Chatting with Your Clock: Using Conversation to Design a Voice User Interface

Adam Coscia
Georgia Institute of Technology
North Ave
Atlanta, GA 30332 USA
acoscia6@gatech.edu

ABSTRACT

Low fidelity in speech recognition technology has resulted in the development of voice user interfaces (VUIs) that prioritize mechanics at the expense of affordance and feedback in their design. In this paper we present a prototype VUI that "converses" with the user by generating novel forms of feedback. The VUI is incorporated into both an alarm and a calendar tool to ground our approach in a common application space. Our system addresses errors through conversation as well, ultimately improving affordances and feedback while minimizing the loss in mechanics. We discuss the potential of this affordance- and feedback- first approach to improve human-machine interactions, culminating in more informed VUI development practices.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General terms: System, Design, Human Factors

Keywords: audio, speech, speech interfaces, speech recognition, conversation, natural language processing, nlp

INTRODUCTION

Recent advances in software capabilities have spurred a growth in ”smart” user interfaces for hardware, such as phones and computers, integrated with an intelligent personal assistant [10]. These software agents take in user commands or questions and perform various tasks and services. One of the most popular forms of assistant, used by over 50% of US adults as of April 2020 [14], is the voice assistant built directly into a hardware's operating system. Several major technology companies have commercialized a version of the intelligent voice personal assistant, including Apple (*Siri*), Microsoft (*Cortana*), Amazon (*Alexa*) and Google (*Google Assistant*). Open-source voice software such as *Mycroft AI* have also been created to allow anyone to develop voice assistant technology for their own projects. The ubiquity, capability and usage of the voice assistant integrated with an operating system reveals a strong desire for user interface design to incorporate speech in near-future systems.

As such, the principles of user interface software underlying the concept of a Voice User Interface (VUI), including **affordances**, **feedback**, and **mechanics** [12], are an important tool for understanding and designing VUIs. VUIs have several advantages when it comes to UI design:

(1) *accessibility* – they only require speech to control them and they can be incorporated into ubiquitous devices, such as smart phones;

(2) *familiarity* – speech is a very well known and utilized form of communication across the world; and

(3) ***mechanics*** – they promote very fast information retrieval and require minimal physical effort.

At the same time, VUIs have several disadvantages as a UI design:

(1) *error rate* – they are prone to frustrating errors, both in the recognition of the user's voice as input, and in the synthesis of human-sounding speech when the system responds to the user's input;

(2) ***affordances*** – they have extremely poor affordances, especially for users that are not tech savvy and may not know what they can do with the interface; and

(3) ***feedback*** – depending on the available speech recognition and synthesis software, VUIs can often have mixed feedback that is not guaranteed to help the user understand what they've done and whether it was successful.

Areas of related research, including natural language processing (NLP), have made strides in processing human speech as text, which in turn can resolve many of the disadvantages brought about when using speech as a UI input. Yet current voice user interfaces, such as those implemented in voice assistants like *Siri*, still prioritize mechanics to overcome errors at the expense of desirable affordances and feedback. This can lead to poor user experiences, discouraging accessibility and representation for diverse users and overall adoption and growth of VUI-integrated technologies. Therefore we ask: how can we design a system that ”converses” with the user to improve affordances and feedback?

In response, we present a conversational VUI that prioritizes affordances and feedback while minimizing the loss in mechanics to provide a more positive user experience. By generating novel forms of feedback, we address errors through conversation. Our interface is integrated with both an alarm application and a calendar application, two common UI application spaces, to ground our approach in real-world scenarios. We posit that our system can drive an affordance- and feedback- first approach to future VUI development, improving human-machine interactions and culminating in more informed VUI development practices.

Our contributions include (1) a conversational VUI prototype (`https://github.com/AdamCoscia/Conversational-VUI`), (2) an understanding of how existing UI frameworks and toolkits can encourage or inhibit the integration of a VUI, and (3) a set of guidelines for developers looking to incorporate VUIs into future applications.

RELATED WORK

UI research has a rich history of exploring how speech can be used to create technology that engages and supports users and their tasks. Here we describe the advancements made in designing speech interfaces and the associated challenges implementing these interfaces.

Speech Interface Design

Progenitors of modern VUI's defined the basic requirements needed for recognizing speech and responding, providing insight into the most important elements of any new VUI. The 1976 Harpy Speech Recognition System [8] used a finite state transition network to store knowledge and searching a network of 1000 words for optimal paths based on a command keyword. Voice Activated Link (VAL), a 1996 precursor to modern dial-in voice-activated menus, featured speech synthesis capabilities for producing auditory feedback to user's spoken commands. [9]. Today, Google's Search By Voice technology, a combination of improved grammar search and speech synthesis, aims for the ubiquity of spoken access by improving availability and performance [15]. The progression of voice technology reveals a desire for robust contextual responses and speech synthesis in a modern, human-centered VUI.

Accessibility issues have necessitated tools which can utilize context to improve interactions. *Capti-Speak* [3], a speech-enabled screen reader and web browser for the visually-impaired, translates speech utterances into browsing actions, executes the actions, and provides audio feedback. Tasks are determined in various ways: in the case of a standard shortcut, *Capti-Speak* consults a pre-defined shortcut-task knowledge base in order to determine the intended task; in the case of a speech utterance, *Capti-Speak* first converts the utterance to text using the Speech Recognizer module (Google ASR) and passes the recognized text on to its Input Interpreter. From the text, the interpreter then extracts information describing the corresponding intended task (action) to be performed and the target object. The interpreter uses personalities that are divided into four categories based on the degrees (high or low) of warmth and competence to determine the appropriate response. Stifleman et al.'s *Audio Notebook* [16] employs *user structuring* and *acoustic structuring* to unburden the user from high cognitive loads while taking notes by engaging with voice as the interaction medium. Moubayed and Lehman [1] investigated how audience, specifically children, can impact design. They designed a speech-controlled robot-child game utilizing Microsoft Speech Analyzer to account for the grammar a child would use to interact with a robot, such as saying 'jump jump jump' instead of just 'jump'. By addressing these and other accessibility concerns, existing design guidelines for VUI development consider the personal relationship the user has with the technology to craft a more engaging experience.

Central to the relationship between the user and machine is how speech is communicated back to the user; i.e., speech synthesis. Yankelovich and Lai [19] classify speech synthesizers along several dimensions. Parameterized synthesizers are small and fast but do not sound very natural. In comparison, concatenative synthesizers sound more natural but also use more resources. Some speech synthesizers need to be trained on a particular speaker's voice, whereas others can be *speaker-independent*. Speech recognizers require the use of a definite grammar, while others allow only certain phrases for control and some are purely based on statistical models. Text-to-speech (TTS) systems [2] translate text into utterances using the various synthesizers above. It is important that synthesizers prioritize *naturalness* and *intelligibility* characteristics in their output [17].

Challenges Interpreting Speech

Implementing speech systems in real-world applications has exposed many issues for designers of VUI's to address, such as incorrect recognition, dialect constraints, limited vocabulary and latency issues [21].

Yankelovich et al. [20] describe several issues in creating VUI's and discuss best practices that should be implemented while designing them: (1) *recognition errors*, such as word error rate; and (2) *nature of speech*, or a lack of visual feedback in speech-only systems that can cause uncomfortable silences for the user while the user is waiting for the device to respond. The first issue, recognition errors, can be further divided into three categories: (a) *rejection errors*, or when recognizers fail to understand an utterance, (b) *substitution errors*, or when recognizers substitute an incorrect word in an utterance, and (c) *insertion errors*, or when recognizers misinterpret noise as an utterance. To minimize rejection errors, the designer should avoid the 'brick wall effect', where every time the recognizer does not understand something, it provides the same response, such as "Sorry, I did not understand that". Rather, the device should have various different responses. In case the recognition of speech fails, it causes an overhead for the user who usually has to stop the conversation and correct the error.

Grammar selection is equally as vital. Speech recognition accuracy depends heavily on the size, coverage, and complexity of the language model used for recognition. Use of domain-specific grammar and vocabulary can be a reasonable choice in order to maximize recognition rates and avoid negative user experiences. Poorly designed grammars may also force users to use unnatural language and make speech recognition tedious to use [18]. A potential solution in speech-based rapid prototyping tools is a threshold that triggers recognition has been set fairly high. These systems strive to avoid error even at the cost of not providing exact feedback [21].

SYSTEM

We built a browser-based client-side VUI integrated with both an alarm and a calendar application using pure *JavaScript*. The source code and a detailed README on how to use and edit the tool can be found in this GitHub repository:

```
https://github.com/AdamCoscia/Conversati
onal-VUI
```

A live demo of the tool can be found here:

```
https://adamcoscia.github.io/sections/pr
ojects/Conversational-VUI/index.html
```

The choice to make the VUI in the browser was motivated by several factors:

(1) *JavaScript* has a rich diversity of robust libraries built for audio input and output (I/O), speech recognition. speech synthesis, and natural language processing (NLP);

(2) the browser is one of the most ubiquitous and accessible interfaces for a wide audience today; and

(3) a client-side browser-based application can be easily hosted and shared for many years with little to no maintenance.

We detail the various facets of the implementation strategy as well as the final build of the tool below.

Design Guidelines

We synthesized various speech interface design guidelines [13, 20, 19] to afford users with the ability to converse with the system. In summary, our system attempts to respond to user requests with a diverse set of possible utterances and different behavior based on conversational context, described below.

Error Handling Our conversational voice user interface strives to avoid three common errors in a traditional speech interface: (1) *insertion error*, (2) *substitution error*, and (3) *rejection error*. We address these common errors in the following ways:

(1) *insertion error* – our system disables the input features of the GUI when the system is processing or speaking so that the user knows it is not currently accepting input;

(2) *substitution error* – our system only asks the user to confirm an action if it would delete or stop showing certain information. This in turn reduces the number of times the user must verify a command; and

(3) *rejection error* – our system offers guidance based on heard utterances; e.g., if the word ”alarm” is heard, but the context of the sentence is lost, the system guides the user towards working with an alarm instead of having the user repeat themselves.

GUI

We created a basic graphical user interface (GUI) to help users debug the application. The GUI presents a few graphical input modalities to help the user begin using the tool. Towards the top of the application are *Voice Settings*, which control the accent, rate, and pitch of the synthesized voice that responds to the user's input. As stated earlier, it is critical that speech synthesizers prioritize *naturalness* and *intelligibility* in their output [17]. We used the *SpeechSynthesis* interface of the Web Speech API [11] (described in Architecture below), which offers a multitude of regional accents to help users interpret the synthesized responses from the system. At any time, the user can press the *Test voice* button to hear a message read aloud by the synthesizer with their current settings.

Below this, the *Start recording* button is used as a switch to allow the browser to begin listening to sounds from the user's microphone and interpret them as human speech. This attempts to mitigate *insertion* errors through a more affordance-first approach. The alternative (i.e., a system controlled solely through voice) cannot resolve such errors. Once pressed, recording will commence immediately and will not stop until the system recognizes that the user has spoken something and then stopped speaking. This is the main input modality for telling the system to take in user input, and the only way for the system to initiate recognition and subsequently synthesize a response.

Finally, the *Show/hide output* button can be pressed to show or hide the area below the button which automatically records the conversation being had as visual text data. This was provided to show that the system can function with and without visual feedback.

VUI

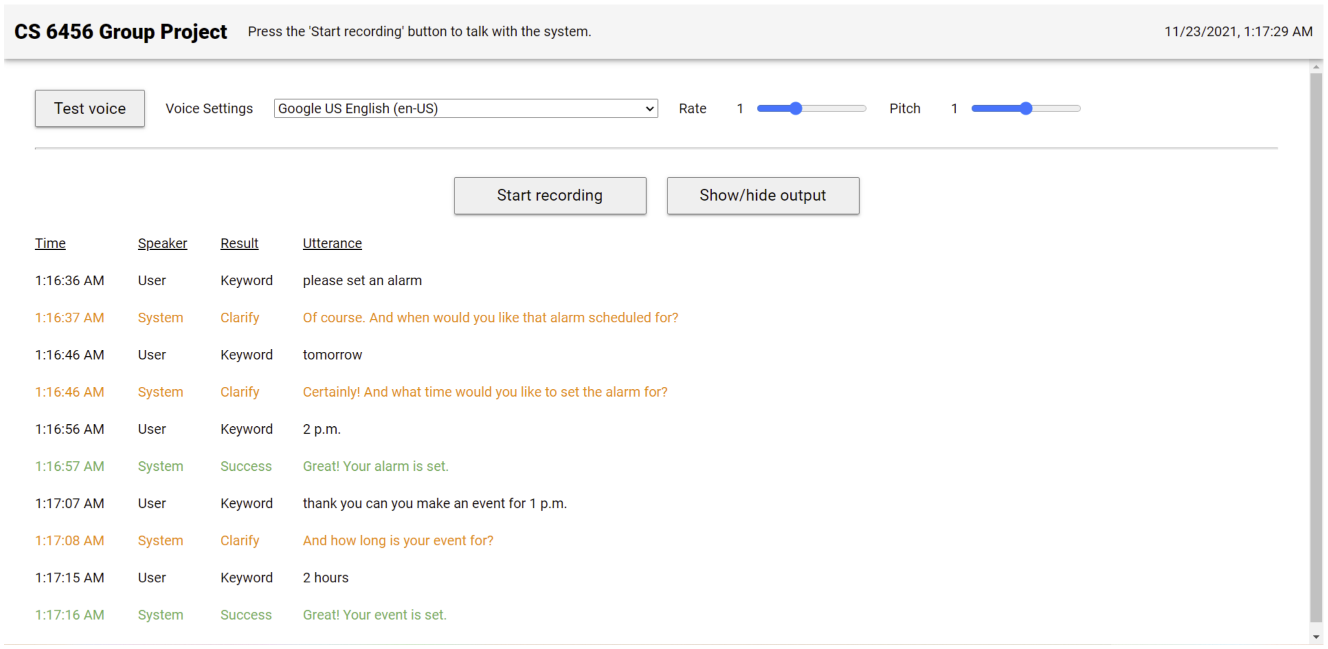
The applications can be interacted with through the voice user interface by pressing the *Start recording* button and speaking aloud. We modeled the tasks a user can perform based on built-in tools widely distributed in both iPhone and Android smart phones and on desktop computers. For example, clock applications, such as Google's *Clock* app [7], let the user work with time management tools such as an *alarm*. Calendar applications, such as Microsoft's *Outlook* app [4], let the user work with date and event tools such as an *calendar*. Using these applications grounds our methodology in a widely used and accessible application space for user interface development.

Alarm Application By pressing the *Start recording* button, the user can begin using the alarm application. We enumerate the operations that users can perform with our alarm application as follows:

(1) *set* – the user can create an alarm that is enabled by default, e.g., by saying ”please set an alarm for Sunday at four a.m.”;

(2) *enable/disable* – the user can enable or disable an alarm without deleting the alarm, e.g., by saying ”please enable my alarm for Sunday at four a.m.”;

(3) *edit* – the user can change the time, date, enabled or dis-


CS 6456 Group Project Press the 'Start recording' button to talk with the system. 11/23/2021, 1:17:29 AM

Test voice | Voice Settings: Google US English (en-US) | Rate 1 | Pitch 1

Start recording | Show/hide output

Time	Speaker	Result	Utterance
1:16:36 AM	User	Keyword	please set an alarm
1:16:37 AM	System	Clarify	Of course. And when would you like that alarm scheduled for?
1:16:46 AM	User	Keyword	tomorrow
1:16:46 AM	System	Clarify	Certainly! And what time would you like to set the alarm for?
1:16:56 AM	User	Keyword	2 p.m.
1:16:57 AM	System	Success	Great! Your alarm is set.
1:17:07 AM	User	Keyword	thank you can you make an event for 1 p.m.
1:17:08 AM	System	Clarify	And how long is your event for?
1:17:15 AM	User	Keyword	2 hours
1:17:16 AM	System	Success	Great! Your event is set.

Figure 1: The graphical user interface for the system. Users can manage synthesized voice settings in the first row as well as provide input to the speech recognition engine and show/hide the output in the second row.

abled status of an existing alarm, e.g., by saying ”please edit my alarm on Sunday to be for Saturday”; and

(4) *delete* – the user can remove an existing alarm, e.g., by saying ”please delete my alarm on Sunday”.

When the exact date and time occurs for an existing alarm that is enabled, the system will play an alarm sound for several seconds to alert the user that the alarm has been triggered. The example phrases are not exhaustive, as the system was built to understand a decently wide variety of input phrases which accomplish the operations above.

Calendar Application By pressing the *Start recording* button, the user can begin using the calendar application. We enumerate the operations that users can perform with our calendar application as follows:

(1) *set* – the user can create calendar events, e.g., by saying ”please set an event on Sunday at four a.m. for 2 hours”;

(2) *edit* – the user can change the date, time, or duration of an existing event, e.g., by saying ”please edit my event on Sunday at four a.m. to be 1 hour long”; and

(3) *delete* – the user can remove an existing calendar event, e.g., by saying ”please delete my event on Sunday”.

The example phrases are not exhaustive, as the system was built to understand a decently wide variety of input phrases which accomplish the operations above.

Architecture

The tool is entirely client-side, browser-based, and written in pure *JavaScript*. The complete list of included libraries and a short of description of their purpose in this project follows.

Web Speech API The Web Speech API [11] is a proposed web standard API that exposes two interfaces, *SpeechSynthesis* and *SpeechRecognition*, which provide speech capabilities in the browser. The *SpeechSynthesis* interface synthesizes human speech in web-based applications. This interface is used to communicate speech back to the user after parsing the user’s input. The *SpeechRecognition* interface is used for speech recognition services in web-based applications. In this project, it is used to understand and interpret as human speech anything that is uttered into the microphone when the system is recording.

SpeechRecognition wrapper *annyang* is a JavaScript library and a wrapper around the *SpeechRecognition* interface of the Web Speech API. It provides utilities for setting handlers and callbacks when recognized speech is passed through the *SpeechRecognition* service. It is used in this project to make writing the various handlers and callbacks much easier.

Audio I/O The *HTMLAudioElement* provides programmatic handling of sound embed in an HTML document. In this project, it is used to play the sound of an alarm clock when a user’s alarm went off.

Topic Modeling and Text Extraction *compromise* is a natural language processing (NLP) library that provides basic document parsing capabilities such as inferring topics, dates, numbers, etc. We used it to get specific features out of the user’s input as text. *compromise* provides several plugins with different specialized functionalities around text parsing. *compromise-numbers* is a plugin that handles parsing and formatting numbers and is required by the *compromise-dates* plugin. *compromise-dates* is a plugin that handles extracting dates from text. We used this plugin extensively to parse the user’s input for dates and times when working with alarms and calendar events.

DISCUSSION

The process of developing a VUI in the browser revealed many opportunities and challenges for working with speech-based systems. We detail several that we encountered for designers and developers to address in their own projects, enumerate various alternatives that were considered for our own work, and conclude with lessons learned.

Challenges

Speech Recognition One of the biggest problems for speech recognizers is simply getting the correct input to be heard by the system. The quality of the speech recognition service played a major role in deciding how we built actions around the recognition. There were several factors that were notably affected by our choice of the Web Speech API:

(1) *volume* – users' speaking volume led to many missed words and phrases. Since we used a laptop computer to test an in-browser VUI, there were limitations in how the computer could be positioned in order to pick up voices around it;

(2) *accents* – testing the system with colleagues that have non-American accents would cause the API to return results with missing or replaced words. This severely affected the affordances of the system and could represent a major area of bias in how speech recognition services are built; and

(3) *colloquialisms* – similarly to accents, colloquialisms were not often picked up the speech recognition service. While it has become a common cultural practice in the US to use formal spoken language when working with voice technology (such as telephone menu systems), the limitations that necessitated such usage only solidify barriers in the way of wider adoption of VUI technology. Representation remains a priority in creating affordance- and feedback- first VUIs.

Even when the system successfully navigated those challenges, there were still issues when using specific phrases. For example, the speech recognition service did not recognize the phrase *"o' clock"* when it was spoken as part of a larger sentence. When this occurred, the recognizer did not provide the system with any indication that this phrase was indeed spoken, even if it did not understand it, leaving it out entirely from the text available to parse. The Web Speech API did not provide as much feedback as we needed to pass on feedback to the end user.

We believe this reveals that the speech recognition service used to build VUIs is one of the most important choices to make as a designer or developer of VUIs. The tools used to create systems heavily affect the outcome. In our case, because of the limitations in parsing volume, accents, and colloquialisms and no way to overcome them, we chose to exclude the use of certain phrases. This ultimately prevented us from providing affordances and feedback in areas they are most needed. We envision several ethical implications in the areas of bias and representation when it comes to future VUI design, and we believe research in this area should focus on mitigating these speech recognition issues.

Text Extraction and Topic Modeling Even when speech can be recognized by a system, there is a lot of post-processing that must happen to determine what the system should do with the spoken text. In particular, numbers and dates were complicated to parse out of text given the vast amount of ways one can express a date and time or an amount of something. When text was determined to be a number or date, context clues from the voice around that value would be needed to correctly associate the spoken value with the subject receiving that value. For example, if a user said *"create an event for three today"* the system cannot disambiguate whether *"three"* is a time or a duration, while a human may implicitly understand that the use of *"three"* is a time. We frequently encountered issues with existing NLP libraries failing to account for this and other difficult linguistic challenges when parsing user input. At the same time, getting the correct subject from a sentence was even more difficult. Topic modeling itself is an open problem in natural language processing (NLP). Unfortunately, we were not able to resolve topic modeling in our system in a consistent way, and thus chose not to include the capabilities in the final build of the prototype.

Lack of Keywords We used the JavaScript NLP library *compromise* to address many of the text extraction and topic modeling issues we encountered. As the name suggests, the library compromises its available lexicon to vastly improve speed and consistency of text parsing under the common assumption that 80% of spoken English consists of only 1000 words. Lack of notable citation aside, we felt this was an incredibly interesting and promising idea to make use of in our work. That is, when parsing numbers, dates, and sentence subjects, we assumed the 80% case of how a user might indicate these things and ignored the 20% of "special cases". This 20% usually fell under the situation when no keywords were able to be parsed in the sentence – nothing about alarms, events, dates, numbers, etc. So, what should happen? This is where NLP can play a major role in the future of VUI development. As the ubiquity, robustness, and interoperability of NLP libraries grows, we should look to incorporate topic modeling and text extraction into speech recognition software. This lowers the barrier of entry for writing VUI-based applications and could result in speech technology breakthroughs.

Design Alternatives

We considered various alternate techniques for designing a conversational VUI that did not make it into the final prototype system.

Conversational Maxims Grice's four maxims of conversation [5] can be used to govern the behavior of a conversational agent:

(1) *be truthful* – e.g., the system should not intentionally report information that is not directly related to internally stored data,

(2) *provide as much information as necessary, but no more than needed* – e.g., the system should deliver only the data requested by the user or perform only the requested action,

(3) *stay relevant to the user's needs/context* – e.g., the system should respond contextually to the user, and

(4) *be clear, concise, and unambiguous* – e.g., the system should provide feedback in the form of short statements that a task has been completed.

Contextual Responses Voice user interfaces can also develop a context around the current "topic" of conversation, in order to afford more streamlined and pleasant interactions with the interface. For example, using pronouns such as "it" to implicitly describe the subject of a sentence based on context around the sentence. We envisioned such interactions requiring an examination of the information flow when transforming GUIs into SUIs [20], as users could easily loose context themselves about what a pronoun is refering to. At the same time, such affordances could vastly improve the experience of using a VUI and even improve the mechanics.

Sub-conversations Consider the back-and-forth nature of conversation. Natural speech patterns often form a series of clarifying questions that helps both participants make incremental progress towards a common understanding without requiring the intermediate steps to be repeated over and over again. In this way, we envisioned that our system could ask for clarification based on recent interactions, or in other words, have a sub-conversation around whatever the current context is. For example, imagine a user asking *"can you set an event for five p.m."*. A conversational system might reply *"you've got a phone call scheduled with Steve then"*. From there, the sub-conversation would allow both the system and user to come to a common understanding – user: *"ok, what about 6?"* – system: *"I can do that!"*. This concept could be replicated across functions beyond scheduling.

Sentiment Analysis Sentences also carry emotional weight to them when spoken aloud. Humans commonly use such emotional "cues" to guide their verbal responses in natural conversation. We envisioned a conversational VUI that could also pick up on such cues and develop responses that appropriately addressed how the user was feeling. For example, if the user was angry and yelled at the system, we would have the system talk briefly with the user about the issue before attempting to continue. NLP already provides facilities to determine various emotional aspects of word choice, phrasing, and delivery. However, text can often fail to convey emotion even when read by humans, and thus we consider that such a solution would ideally be built directly into the speech recognition service.

Implementation Alternatives

We also considered alternative applications that could work in this space as well. These were not included in the final build.

Note-taking application Note-taking applications, such as Apple's *Notes* app [6], let users work with text data. Considering the ubiquity of text-related operations that users can perform using graphical user interfaces, we wanted to explore the ability for our interface to replicate similar tasks using voice. We categorized the tasks users would have been able to accomplish as follows:

(1) *write* – the user can dictate individual notes to be recorded as text,

(2) *group* – the user can group these notes into single-level lists,

(3) *search* – the user can ask for the containing list(s) or note(s) that match the conditions of a search phrase, and

(4) *read* – the user can request audio playback of any note.

Lessons Learned

From our system, we learned several lessons that should be taken into careful consideration when designing future systems.

Expect the unexpected. Elements of VUI design evolve rapidly: grammar (including accents and colloquialisms); prosody (sentiment analysis); speech recognition services (and associated APIs); speech synthesis services; natural language processing; etc. Building code to handle such edge cases in an affordance- and feedback- first way makes for a more robust and rewarding system for users.

Your choice of API matters. As discussed in the challenges with speech recognition section, the inability to debug how speech was being recognized inhibited us from parsing certain phrases. It is vital that designers and developers give due consideration to the quality of the recognition service they incorporate in their system. This only further cements the idea that...

Representation matters. Systems will only be effective insofar as they represent the user's voice. There are many ethical implications that deserve consideration, including bias in how recognizers respond to accents and colloquialisms, accessibility for diverse audiences, etc.

Look to other areas for inspiration. By thinking about how to incorporate ideas from a broader spectrum of contexts, developers can often find inspiration and use cases from related libraries. In our case, NLP revealed many new ideas for interpreting voice by using text.

Build for the 80% use case. Making compromises in mechanics can improve the affordances and feedback of your system tremendously.

CONCLUSION

Our conversational VUI prototype prioritizes various affordances and feedback methods while attempting to minimize the loss in mechanics throughout the user experience. Development of our system revealed that there are several major challenges involved with speech recognition, text extraction, topic modeling, and a lack of keywords. Investigating these issues revealed opportunities to consider accessibility, bias, and representation in future work on speech recognition services and the VUIs that adopt them. These results can have far-reaching implications for the designer or developer looking to create novel VUI-based technology.

REFERENCES

1. Samer Al Moubayed and Jill Lehman. Design and architecture of a robot-child speech-controlled game.

In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction Extended Abstracts*, pages 79–80, 2015.

2. Jonathan Allen, M Sharon Hunnicutt, Dennis H Klatt, Robert C Armstrong, and David B Pisoni. *From text to speech: The MITalk system*. Cambridge University Press, 1987.

3. Vikas Ashok, Yevgen Borodin, Yury Puzis, and IV Ramakrishnan. Capti-speak: a speech-enabled web screen reader. In *Proceedings of the 12th International Web for All Conference*, pages 1–10, 2015.

4. Microsoft Corporation. Outlook web app, 2021. `https://outlook.com/`.

5. Herbert P Grice. Logic and conversation. In *Speech acts*, pages 41–58. Brill, 1975.

6. Apple Inc. Notes, 2021. `https://apps.apple.com/us/app/notes/id1110145109`.

7. Google LLC. Clock, 2021. `https://play.google.com/store/apps/details?id=com.google.android.deskclock`.

8. Bruce T Lowerre. *The harpy speech recognition system.* Carnegie Mellon University, 1976.

9. M Manjutha, J Gracy, P Subashini, and M Krishnaveni. Automated speech recognition system–a literature review. *COMPUTATIONAL METHODS, COMMUNICATION TECHNIQUES AND INFORMATICS*, page 205, 2017.

10. Karen Myers, Pauline Berry, Jim Blythe, Ken Conley, Melinda Gervasio, Deborah L McGuinness, David Morley, Avi Pfeffer, Martha Pollack, and Milind Tambe. An intelligent personal assistant for task and time management. *AI Magazine*, 28(2):47–47, 2007.

11. André Natal, Glen Shires, and Philip Jägenstedt. Web speech api, 2020. `https://wicg.github.io/speech-api/`.

12. Don Norman. *The design of everyday things: Revised and expanded edition*. Basic books, 2013.

13. Charles L Ortiz. The road to natural conversational speech interfaces. *IEEE Internet Computing*, 18(2):74–78, 2014.

14. Edison Research. The smart audio report. *National Public Media*, Apr 2020.

15. Johan Schalkwyk, Doug Beeferman, Françoise Beaufays, Bill Byrne, Ciprian Chelba, Mike Cohen, Maryam Kamvar, and Brian Strope. *"Your Word is my Command": Google Search by Voice: A Case Study*, pages 61–90. Springer US, Boston, MA, 2010.

16. Lisa Stifelman, Barry Arons, and Chris Schmandt. The audio notebook: paper and pen interaction with structured speech. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 182–189, 2001.

17. Paul Taylor. *Text-to-speech synthesis*. Cambridge university press, 2009.

18. Markku Turunen, Aleksi Kallinen, Ivàn Sànchez, Jukka Riekki, Juho Hella, Thomas Olsson, Aleksi Melto, Juha-Pekka Rajaniemi, Jaakko Hakulinen, Erno Mäkinen, et al. Multimodal interaction with speech and physical touch interface in a media center application. In *Proceedings of the International Conference on Advances in Computer Enterntainment Technology*, pages 19–26, 2009.

19. Nicole Yankelovich and Jennifer Lai. Designing speech user interfaces. In *CHI 98 Conference Summary on Human Factors in Computing Systems*, pages 131–132, 1998.

20. Nicole Yankelovich, Gina-Anne Levow, and Matt Marx. Designing speechacts: Issues in speech user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 369–376, 1995.

21. Roman Zenka and Pavel Slavik. Supporting ui design by sketch and speech recognition. In *Proceedings of the 3rd annual conference on Task models and diagrams*, pages 83–90, 2004.